To BDI, or Not to BDI: Design Choices in an Agent-Based Traffic Flow Management Simulation

Shawn R. Wolfe
NASA Ames Research Center
Shawn.R.Wolfe@nasa.gov

Maarten Sierhuis
RIACS/NASA Ames Research Center
Maarten.Sierhuis-1@nasa.gov

Peter A. Jarvis
Perot Systems Government Services/NASA Ames Research Center
pjarvis@email.arc.nasa.gov

Abstract

Belief-Desire-Intention (BDI) is a powerful agent paradigm that allows for the development of so-called intelligent agents – agents that can reason and act based on their beliefs and intentions. However, this power often comes at the cost of increased computational overhead. We describe our experience using a BDI agent framework for developing a simulation of collaborative air traffic flow management and the efficiency problems we encountered. By using BDI more judiciously in our simulation, we were able to address these issues and greatly reduce the execution time of our simulation. From our successes and failures, we derive several guidelines that may enable other researchers to avoid similar efficiency issues in BDI-based simulations.

1. Introduction

The National Airspace System (NAS) of the United States today is under the control of the Federal Aviation Administration (FAA). Aided by their more comprehensive situational knowledge, the FAA's air traffic controllers work with pilots to ensure safety in the airspace. Though pilots of small general aviation (GA) aircraft are often able to fly safely using visual flight rules (VFR) and minimal controller direction, pilots of larger commercial aircraft rely heavily on air traffic controllers, as they have little visual warning at the correspondingly higher flight speeds. Air traffic flow management, then, is primarily focused on managing these commercial flights safely through traffic congestion, poor weather and other potential hazards.

To make this task manageable for the controllers, commercial aircraft generally follow structured air traffic routes, essentially "highways in the sky". These air routes greatly increase the predictability and manageability of traffic flows, but at the cost of freedom of movement. Though not readily observable to the naked eye, aircraft are often queued up in these flight routes, much like cars on a busy freeway. Unlike those cars, however, aircraft must operate within a narrow range of flights speeds to avoid stalling or structural damage. These operating constraints make the traffic flow management task more challenging, reducing the number aircraft the controllers can manage. A forecast of air traffic demand in 2025 shows an increase of two to three times over present day levels [1], resulting in even greater flight delays.

The Next Generation Air Transportation System (NGATS) project is a multi-faceted research effort to address issues with the NAS. One such facet is the area of Collaborative Traffic Flow Management (CTFM), which intends to increase both the efficiency of the NAS and the satisfaction level of the airlines. In today's system, the flow of traffic is primarily handled by three entities: the FAA's Air Traffic Control System Command Center (ATCSCC) and, Traffic Management Units (TMUs), and the individual airlines' Airline Operation Centers (AOCs). Previous field observations found that several aspects of the current system hindered collaboration [2]. A new concept of operations was suggested to address these issues [3], and our goal is to evaluate this concept through an agent-based simulation.

We begin (in Section 2) with related TFM simulations, categorizing them within or outside of the agent-directed simulation taxonomy. In Section 3 we state the problem we are trying to model and simulate (i.e. a specific traffic flow problem in the NAS), and in Section 4 we motivate our use of the BDI agent paradigm and characterize the simulation platform (Brahms) used. The primary contribution follows in the remainder of the paper, as we describe an idealized,

naïve model design that incorporates earlier flaws, and contrast this naïve design with our later design. From this experience, we derive general guidelines for the use of BDI agents in large-scale simulations and conclude with future work.

2. Related TFM Simulations

TFM involves both complex physical processes (e.g., aerodynamics) and complex human systems (e.g., coordinated action and distributed decision making). An earlier focus on physics-based modeling has lead to several excellent simulators of the first type, and so increasingly the focus has been shifting to the simulating the roles of people in the TFM system. The use of the agent paradigm is growing in TFM simulations, but the use of BDI agents in TFM has yet to enjoy widespread adoption.

Agent-directed simulation separates the use of agents in simulation into the category of *simulation for agents* and *agents for simulation* [4]. In *simulation for agents,* simulation techniques are used for simulating real-world entity behavior, e.g. simulation of cognitive behavior. *Agents for simulation* itself is divided into two categories: *agent-supported simulation,* the implementation of a simulation environment with the help of agent technology; and *agent-based modeling and simulation* (ABMS), using interacting software agents modeled to generate emergent system behavior.

Simulation for Agents of TFM: The Airspace Concept Evaluation System (ACES) [5] is a distributed agent simulation of the NAS. ACES is based on the Department of Defense's High Level Architecture (HLA), which has enabled the integration of several simulations into the overall system. As ACES is focused on the entire NAS, the simulation includes traffic flow management [6], but is not specifically focused on TFM.

The Man-Machine Integrated Design and Analysis System (MIDAS) [7] is another example of an agent simulation. MIDAS is a human performance simulator for pilots or flight controllers, focusing on the limitations of cognitive ability more than the results of complex decision making.

Agents for Simulation of TFM: IMPACT (Intelligent agent-based Model for Policy Analysis and of Collaborative Traffic flow management) uses an ABMS approach to simulate the interaction between simple agents in an economically based environment [8]. In the simulation, policy-based FAA agents evaluate and impose ground delay programs (GDP), based on the capacity of the airspace and the weather. Their decisions are based on simple rules about capacity of airports and equality between airlines. The airline agents are economically based agents and make their decisions based on calculated cost. By imposing specific random events at the start, the output of the simulation are statistics based on the emergent agent behavior in the system. The purpose of the IMPACT simulation is very similar as the purpose of our simulation. However, our agent approach differs in that IMPACT models airlines and FAA as swarm-based agents that use a simplistic decision-making algorithm. In contrast, we use a more complex agent model of the organization of both airlines and FAA, using BDI agents that can reason and communicate with other agents in the model.

Tumer and Agogino [9] use FACET (see below) to test a multi-agent algorithm for traffic flow management. They use a Monte-Carlo simulation to estimate the congestion within a certain traffic management unit (TMU) within the NAS, based on simple agents' actions to speed up or slow down traffic. The agents use reinforcement learning to set the separation between airplanes to manage congestion.

Whereas our simulation focuses on the collaborative decision-making process between the airlines and FAA before flights take off, Tumer and Agogino focus on adaptive agents taking independent actions that maximize a system evaluation function for enroute flights. They use an ABMS approach to evaluate their AI-algorithm. Again, agents are very simple computational objects representing fixes in a 2D space, and do not have any similarities with entities in the real world (e.g. particular people in a FAA control center).

Non-Agent Simulations of TFM: Hogan and Wojcik [10] simulate a "day in the life" of Newark airport. Their simulation model is not agent-based: although their paper does not provide a description of the model representation, it is clear that airport, runways, routes, etc, are at most represented as structured records or objects. Decision-makers are not modeled in any detail.

FACET [11] [12] is a NASA-developed tool for simulating air traffic flow. FACET contains modules that concentrate on trajectory modeling, weather modeling, and also contains a model of the airspace structure, including the ARTCC regions, sectors, and air routes. FACET can act either as a simulator or as a playback mechanism, using either from historical data or from a live data feed from the FAA. FACET has been integrated into a commercial product, Flight Explorer [13] which is used by the majority of major U.S. airlines. FACET is not an agent-based simulation, concentrating primarily on the physical aspects of air traffic flow, but does include non-physical aspects such as controller workload and air traffic management initiatives.

3. Simulating CTFM

The CTFM concept of operations is quite complex, requiring a multi-year effort to develop a complete simulation. Our initial efforts focused only on a subset of the CTFM concept of operations: the route selection problem [14]. We also made several simplifications to speed up the initial modeling task. In reality, each controller controls a three-dimensional space known as a sector, but in our model we assigned one controller per route and did not include sectors. The controllers themselves were modeled only as a constraint, i.e., the number of flights that could follow a particular air route, which we considered as a route capacity. The designs presented in this paper do not include persons or roles within an organization, as our BDI agents correspond to the organizational level. Therefore, an AOC or TMU is modeled as a single agent rather than a more complex set of cooperating "people" agents. A more complex model that captures individual roles is currently under development but falls outside of the scope of this paper.

We created several simple traffic scenarios with varying levels of traffic. All scenarios used traffic between seven airports, three airlines, and controlled by only one TMU. This meant that we had only one TMU agent and three AOC agents, and that we did not need to model TMU to TMU interactions. Three air routes were created between each airport pair: a primary route, preferred by all airlines, and two less desirable secondary routes. Though the capacity on the primary route was not always enough for all scheduled traffic, there was enough capacity amongst the three routes to accommodate all scheduled traffic. Our goal was to measure how effectively routes were assigned to flights by either the TMU or the AOC.

We created several variants of the simulation to provide points for comparison. In the first variant, the TMU chose all routes for all flights without input from the AOC or concern for the relative value of each flight: this simulation roughly corresponds to current operations. In the second variant, the TMU again chose routes for all flights, but would use the value of the flight and a greedy algorithm to reach a globally optimal solution: this simulation captured the best performance possible, according to our metric. In the third variant, the AOCs would choose routes themselves in an iterative process, with the TMU evaluating the global solution and notifying the AOCs of any constraint violations. This variant most closely corresponded to the CTFM concept of operations. In all cases, we ignored the possibility of weather and other such disruptions; the only constraint was the limited (but static) capacity of the routes.

4. The Case for BDI

Our notion of modeling collaboration between human organizations is based on our *work practice simulation approach* [15]. Although modeling *work practice* based on a *concept of future collaboration* between people is not possible (as the work practice has not been established), the agent-based modeling approach we developed for modeling work practice in organizations allows us to instantiate the concept process flow in a realistic agent-based model. Such an ABM enables the investigation of how people will actually collaborate in the future process.

The main representational paradigm for BDI agents is declarative antecedent-consequence rules, similar to the old rule-based expert systems [16] [17]. Each BDI agent can be seen as a knowledge-based system that represents the reasoning capability of a particular agent. Therefore, a multi-agent BDI simulation includes a number of parallel-executed knowledge-based agents that represent people performing particular roles in organizations. We use BDI agents to model the collaborative decision making in each phase of the process flow. This allows us to model the collaborative decision making at a realistic level of people working together in an organization. This is in contrast to the simple algorithmic agents used in [8], [9] and [10].

The power of the BDI approach is that we can represent how collaborative decision making can happen in tomorrow's organizations of airlines and FAA. We represent the work activities of the different roles needed to implement the CTFM concept process.

We use the Brahms multi-agent simulation language [18] to model the concept collaborative process flow described by Garcia-Chico et al [19]. Brahms is a tool for modeling and simulating the way people work and collaborate, and use systems to accomplish their tasks. Brahms agents are both BDI agents, as well as subsumption-based reactive agents (see Figure 1) [20]. Brahms can be used to describe current and future work processes and practices in human and other types of organizations. Another application of Brahms is to design the collaborative activities between multiple intelligent agents— both human and software agents [21].

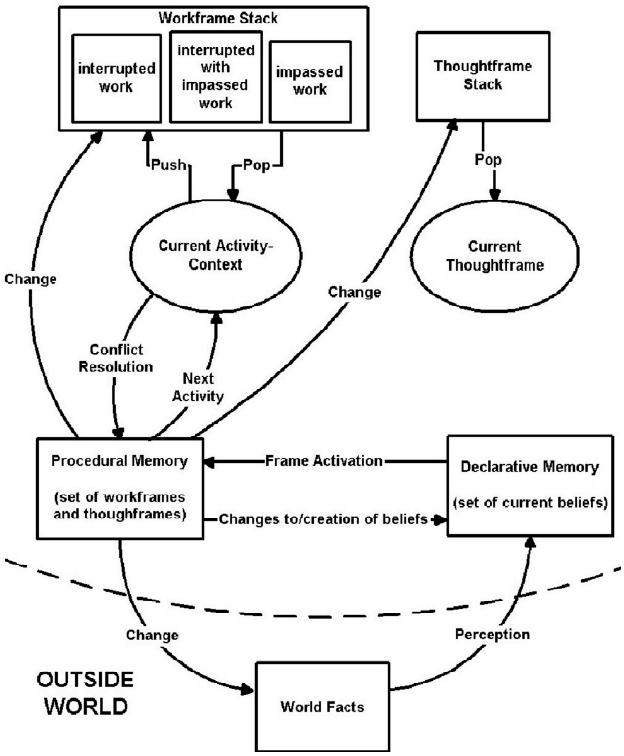

Figure 1. Brahms Agent Architecture

Figure 1 shows the architecture of a Brahms agent. Each Brahms agent runs as a separate Java thread within the discrete-event Brahms Virtual Machine. A Brahms agent has a belief-driven inference engine that continuously monitors the changes and/or creation of beliefs that occur in the agent's declarative memory. Every belief change triggers an evaluation of possible frame activations in the agent's procedural memory. The procedural memory consists of two types of frames (i.e. rules): workframes and thoughtframes. Thoughtframes are forward-chaining production rules that take no simulated time to fire. Workframes, on the other hand, are forward-chaining rules that call activities that take simulated time. Consequently, workframes take simulated time to execute.

With every discrete-event change in the agent's declarative memory, a number of workframes and/or thoughtframes have the potential of firing. These become part of the *workframe-* and *thoughtframe stacks*. The *thoughtframe stack* uses a simple thought frame priority schema to select the *current thoughtframes* to fire in the current clock tick. In contrast, the *workframe stack* has a complex conflict resolution schema that selects the agent's *current activity*, using a workframe and activity state-transition model.

Finally, Brahms allows for modeling the environment and its state as facts in the *World Facts Model* (WFM). Through a reactive method, agents can *detect facts* in the world state, modeling an agent's perception. Detected facts become beliefs added to the agent's declarative memory. However, fact detection is activity-context specific so that not every fact detected by the agent. Agents, through the execution of actions in the world can also change the state of the world by creating or changing facts in the WFM. A more complete explanation of how Brahms works can be in found in [15].

5. Naïve Design

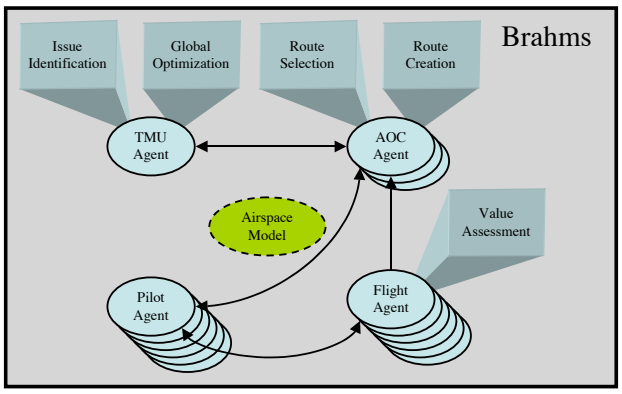

Figure 2. Naïve Simulation Design. *All simulation components are implemented in the Brahms language. Blue ovals represent agents, with stacks abstractly representing the relative number of instantiations. Blue 3-D boxes represent significant reasoning modules. Green dashed ovals represent complex state representation.*

In our naïve simulation design, we implemented all the simulation components directly in the Brahms language (see Figure 2). Along with several minor agents, the simulation had four main agent types:

1. **Pilot Agents.** As the simulation began, each pilot agent would report for duty with their corresponding AOC and transmit flight schedule information (e.g., scheduled takeoff time, destination and arrival times) to the AOC. They would receive flight route assignments from the AOC when such were available, and fly the aircraft along this route at the scheduled time.
2. **Flight Agents.** Though not deliberative agents, flight agents were created and given several bookkeeping tasks, like reporting position and calculating flight value information to the AOC.
3. **TMU Agent.** The TMU agent served as the monitor of the airspace, detecting demand-capacity imbalances and broadcasting such information to the AOCs. In the primary variation of the simulation, the TMU would accept or reject flight route requests from the AOC; in the other simulation variations, the TMU agent would

assign routes instead of the AOCs, using either an optimal or suboptimal strategy.

4. **AOC Agents.** The AOC agents are the airlines' interface to the FAA and communicate information such as flights, schedules, and flight value information, as well as receive any transmitted information from the TMU. In the primary simulation variant, the AOCs would select routes for their flights, re-planning whenever those route selections were rejected by the TMU; in the other variants the AOCs had no direct role in route selection. In an earlier version of the simulation, the AOCs also created the flight routes themselves, using a heuristic search to combine path segments into a reasonable route to the desired destination.

In addition to these agents, we also represented aspects of the environment (i.e. the airspace) as facts in Brahms, necessarily, so that the agents can detect facts in the environment and reason about them. Therefore, physical quantities such as waypoint position and route lengths were directly represented in the Brahms simulation, along with incorporeal properties like controller workload and impacted areas.

Unfortunately, it was quickly apparent that the naïve design would not yield a usable implementation. Though there was no explicit run time performance requirement, an excessively slow simulation would lead to difficulties with debugging and further development. Even with only a partial implementation, simulation runs were taking tens of minutes, which was a real concern given the relatively low level of complexity achieved.

6. Revised Design

We re-evaluated our design choices in order to address the efficiency problems we encountered. In this redesign, we considered two issues: what we had chosen to simulate, and how those components could be simulated.

6.1. Simulation Simplifications

Our focus in the initial simulation was to study the route selection problem in collaborative and authoritarian settings. Upon further reflection, it was clear that the essential component was really the advanced planning, and not the execution of the plan itself. Correspondingly, we sought to simplify the model by excluding portions that had real world counterparts, but were not pertinent to our current focus.

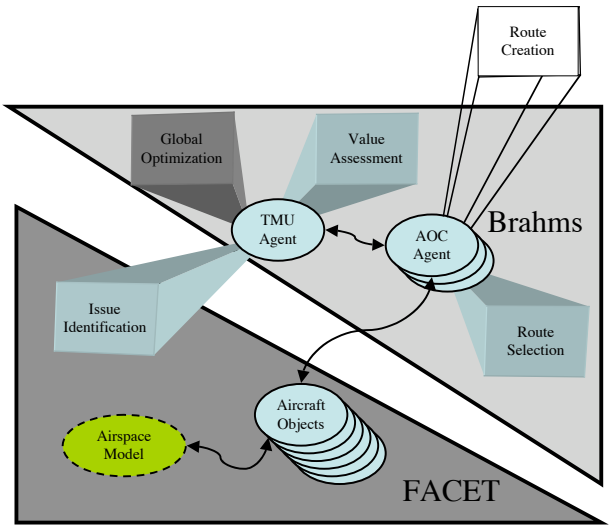

Figure 3. Simulation Redesign. *By simplifying, removing or redistributing simulation components, we were able to reduce the run time of the simulation to acceptable levels.*

We removed the Brahms pilot agents, as they did not play an integral role in the planning phase, and changed the flight agents to passive flight objects. The bookkeeping and flight value computations were moved from the flight to the TMU, partially for efficiency concerns, but also to avoid the situation where an AOC might artificially inflate flight values to get better treatment. Had we decided to preserve the execution phase, we still could have eliminated our pilot agents, as they robotically followed the commands of the AOC and were not enriched by the BDI representation of Brahms.

Similarly, the route creation fell outside of the planning phase, as a prior process that creates routes as inputs to planning. Instead of creating routes on the fly, we structured fixed routes before the simulation started. This meant that we only had to create the routes once, rather than every time we ran the simulation.

6.2. Implementation Choices

The remaining components were vital to the simulation and could not be eliminated. However, not all components were justified in their use of a BDI model. We had implemented a greedy algorithm to create a globally optimal selection of routes for flights across all airlines. The algorithm had not been designed to model the actual process a human would follow to reach this globally optimal solution; it was merely created to serve as a point of comparison. As such, it was not necessary or even logical to implement this function in a BDI framework; only the resulting solution was needed. Therefore, we used

Brahms capability to interface with Java[1] to implement the algorithm.

The other major implementation change was to rely on the simulation capabilities of FACET (see Section 2) for many of the simulated components. Through the use of Brahms Java agents and the FACET Java API, we were able to utilize the optimized simulation capabilities of FACET for the airspace environment model instead of our own implementation. However, the agents in our simulation do reason over aspects of the airspace; this required a representation of the salient features in the BDI framework. Great care was taken to only represent the airspace components that were truly needed for agent decision making, and such components were represented in the most abstract (i.e., compact) form possible. In some cases, such as the identification of the demand-capacity imbalance, the processing was done in FACET (and supporting Java code) and only the outcomes were represented in the corresponding agent's belief model.

Our implementation of the resulting redesign was several times faster than our partial implementation of the naïve design. Not only did our revised implementation include components that our initial implementation lacked, but also did so at a higher level of fidelity, due to our use of FACET.

7. Discussion

The inefficiencies of our initial design where primarily caused by our overuse of the BDI paradigm in the simulation. The choice to model both pilots and flights as agents meant that there would be hundreds of such agents, far more than all other agents in the simulation combined. In Brahms, like many other BDI-based agent languages, each agent corresponds to a separate processing thread [22]. Each agent must continuously monitor and process events that trigger a reaction (e.g., detection of a relevant facts in the world state or communication of beliefs by other agents), slowing the process considerably (also noted incidentally in [23])

Pilots and flight agents were not essential to our simulation goals and were somewhat easily eliminated, but other elements were vital (such as a model of the airspace and optimization routines). However, we realized that we had resorted to using the BDI paradigm as if it were an imperative programming language. Outside of programming convenience, we could not justify the use of the BDI paradigm to develop models for "non-thinking" entities such as the airspace and airplanes [24]. Also, it made little sense to model complex decision making with the BDI framework when we were neither interested nor informed of the actual steps of the process.

In our experience, the most effective approach for most simulation is thus to combine BDI agents and non-BDI components to satisfy simulation requirements. We suggest the following guidelines to address efficiency issues, in our order of decreasing preference:

1. **Use BDI for explicit cognitive processes**. The BDI paradigm is an excellent choice when modeling well-understood decision making. On the other hand, other simulation techniques are better suited for modeling physical and likewise processes that are not analogous to the reasoning supported by BDI. Also, decision-making processes whose details are not understood are not well served by a full BDI implementation, as we observed in our optimized route selection process. In such cases an algorithmic approach, as opposed to a model-based symbolic approach is often more efficient.
2. **Scope the simulation carefully.** Including elements in the simulation that do not support simulation goals can adversely affect the simulation. When building a simulation, there is a natural tendency to include whatever entities are involved in the real world, so care must be taken when choosing what to model. In addition, the level of granularity for modeling processes is important; there is no need to model the details of a decision-making process when you are only concerned with the outcome, rather than the process itself. Careful scoping is a general principle independent of efficiency. However, it is especially important to reconsider functionality that is causing serious performance problems.
3. **Consider the execution properties of the language.** Every BDI implementation eventually turns into code executing on a computer, and understanding the algorithmic properties is sometimes necessary. In Brahms, we found that a straightforward expression of multiple preconditions can lead to a combinatorial explosion in execution time; by considering how the underlying simulation engine executed it, we were able to create a logically equivalent form with vastly superior performance. Similarly, as performance degrades with the number of agents introduced, it is sometimes beneficial to combine processes from multiple agents into a single agent (see Figure 4). However, since these techniques generally decrease readability and correspondence with the real world, we recommend using them only as a last resort.

[1] There are two ways to interface with Java in Brahms. Both make use of the extensive Java API: 1) Java activities are agent actions implemented in Java, 2) Java agents are "agentified" Java objects implemented completely in Java that other Brahms agents can communicate beliefs with. Brahms Java agents do not have an inference engine as shown in Figure 1.

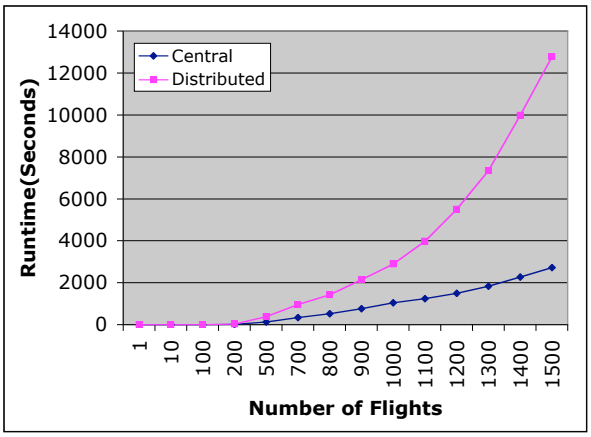

Figure 4. Flight Value Computation Efficiency. *Using distributed agents to calculate flight value resulted in an exponential increase in computational time as the number of flights increased. The same operation, calculated centrally, had superior performance characteristics.*

8. Future Work

Efficiency is likely to remain a concern as we expand the simulation to more accurately reflect the proposed CTFM concept. Consider the fact that to simulate the entire NAS the simulation needs to deal with more than 50,000 flights per day, 26 TMUs organizations and tens of AOCs. We believe that the guidelines presented in this paper will help us avoid many of the potential efficiency problems we might otherwise encounter. Specifically, we plan to expand our simulation in several ways:

1. **Expand simulation scope.** Our initial simulation efforts focused entirely on the TFM planning phase, but the complete CTFM concept covers several phases that influence each other. As we shift to this multiphase model, the dynamism in the airspace environment will play a vital role. Fortunately, by integrating FACET into our simulation, we will be able to make use of a highly optimized airspace simulator. Our design challenge changes from creating an airspace simulator to deciding how to efficiently transmit information between the two simulations. Most likely, careful scoping will be necessary.
2. **Increase decision making fidelity.** Initially, we have only modeled simple decision-making processes in our simulation. In reality, the decision-making processes followed by the corresponding real life entities are neither simple, well understood nor publicly available. Accurate modeling of decision making is necessary for overall simulation fidelity and remains a challenge. One possibility is to use historical data and learning techniques to induce a model. In any case, since we are neither likely to learn the detailed steps of such decision making, nor are we interested in validating that process, we will model such processes at a large-grained level without using BDI internally to arrive at the resulting decision.
3. **Increase simulation size.** TFM issues range from those local to a sector or airport to ones that are truly national in scope. Even at the local level a major airport will have hundreds of flights from tens of different carriers. To date, our simulations have included only a handful of agents (as measured in our redesign), but we must scale up to a much larger number to simulate other types of TFM issues. There is a "breaking point" in every simulation were it is no longer feasible to increase the size of the simulation. This point may come earlier in BDI systems that use a sophisticated level of processing for each agent. One possibility, supported by Brahms, is to distribute the simulation amongst several computers, though this introduces new issues of complexity and limited hardware. In the end, though, it may not be feasible to create a full-sized simulation of CTFM involving thousands of BDI agents using existing simulation packages such as Brahms.
4. **Modeling of Organizations.** We will increase the fidelity of the simulation of the human decision making by modeling the roles and role interaction in the different organizations, both for TMUs and AOCs.
5. **Systematic Performance Investigation.** A thorough empirical study of the performance characteristics of the different design choices would yield quantitative results that may inform future design decisions. Specifically, we intend to run a series of experiments that compare the run time properties of different options. One possible experiment is to contrast the efficiency of a complex algorithm implemented in a BDI framework and the same algorithm implemented in a procedural language (such as Java).

9. Acknowledgements

The authors would like the NGATS ATM-Airspace project for their support of this work, as well as Francis Enomoto, Dr. Karl Bilimoria and Bart-Jan van Putten for their contributions to the work and comments.

10. References

1. Federal Aviation Administration, *FAA Long-Range Aerospace Forecasts Fiscal Years 2015, 2020 And 2025*. 2000.
2. Idris, H., et al. *Field Observations of Interactions Between Traffic Flow Management and Airline Operations*. in *AIAA 6th Aviation, Technology, Integration, and Operations Conference (ATIO)*. 2006. Wichita, Kansas.
3. Idris, H., et al. *Operational Concept for Collaborative Traffic Flow Management based on Field Observations*. in *AIAA 5th Aviation, Technology, Integration, and Operations Conference (ATIO)*. 2005. Arlington, Virginia.
4. Yilmaz, L., T. Ören, and A. Nasser-Ghasem, *Agents, Simulation, and Gaming*. Simulation and Gaming Journal, 2006. **37**(3): p. 339-349.
5. Sweet, D.N., et al. *Fast-Time Simulation System for Analysis of Advanced Air Transportation Concepts*. in *AIAA Modeling and Simulation Technologies Conference and Exhibit*. 2002. Monterey, California.
6. Couluris, G.J., et al. *National Airspace System Simulation Capturing the Interactions of Air Traffic Management and Flight Trajectories*. in *AIAA Guidance, Navigation, and Control (GNC) Conference*. 2003. Austin, Texas.
7. Corker, K.M., *Human Performance Simulation in the Analysis of Advanced Air Traffic Management*, in *1999 Winter Simulation Conference*, P.A. Farrington, et al., Editors. 1999.
8. Keith C, C., et al., *Modeling Distributed Human Decision-Making in Traffic Flow Management Operations*, in *3rd USA/Europe Air Traffic Management R&D Seminar*. 2000: Napoli, Italy.
9. Tumer, K. and A. Agogino, *Distributed Agent-Based Air Traffic Flow Management*, in *AAMAS 2007*. 2007: Honolulu, Hawaii. p. 330-337.
10. Hogan, B. and L.A. Wojcik, *Traffic Flow Management Modeling and Operational Complexity*, in *2004 Winter Simulation Conference*, R.G. Ingalls, et al., Editors. 2004.
11. Bilimoria, K., et al. *FACET: Future ATM Concepts Evaluation Tool*. in *3rd USA/Europe ATM 2001 R&D Seminar*. 2000. Napoli, Italy.
12. Smith, P., et al., *The design of FACET to support use by airline operations centers*. 2004, IEEE.
13. *Flight Explorer*, Flight Explorer Inc.: http://www.flightexplorer.com/.
14. Wolfe, S.R., et al. *Comparing Route Selection Strategies in Collaborative Traffic Flow Management*. in *Intelligent Agent Technology (IAT 2007)*. 2007. Fremont, CA, USA: IEEE press.
15. Sierhuis, M., W.J. Clancey, and R.v. Hoof, *Brahms: A multiagent modeling environment for simulating work processes and practices*. International Journal of Simulation and Process Modelling, Inderscience Publishers, 2007. **3**(3): p. 134-152.
16. Clancey, W.J. *The advantages of abstract control knowledge in expert system design*. in *AAAI-83*. 1983.
17. Clancey, W.J., *Heuristic Classification*. Artificial Intelligence, 1985(27): p. 289-350.
18. Clancey, W.J., et al., *Brahms: Simulating practice for work systems design*. International Journal on Human-Computer Studies, 1998. **49**: p. 831-865.
19. Garcia-Chico, J.-L., et al., *Collaboration for Mitigating Local Traffic Flow Management (TFM) Constraints due to Weather, Special Use Airspace (SUA), Complexity, and Arrival Metering: Prototype Algorithms for Collaborative TFM*. 2007, L-3 Communications and Metron Aviation: Billerica, MA
20. Sierhuis, M., *"It's not just goals all the way down" – "It's activities all the way down"* in *LNAI 4457: Engineering Societies in the Agents World, Seventh International Workshop (ESAW'06)*. 2007, Springer-Verlag: Dublin, Ireland. p. 1-24.
21. Sierhuis, M., *Modeling and Simulating Work Practice; Brahms: A multiagent modeling and simulation language for work system analysis and design*, in *Social Science Informatics (SWI)*. 2001, University of Amsterdam, SIKS Dissertation Series No. 2001-10: Amsterdam, The Netherlands. p. 350.
22. Bordini, R.H., et al., eds. *Multi-Agent Programming: Languages, Platforms and Applications*. Multiagent Systems, Artificial Societies, and Simulated Organizations, ed. G. Weiss. 2005, Springer Science+Business Media, Inc.: New York, NY.
23. Thangarajah, J., L. Padgham, and J. Harland, *Representation and reasoning for goals in BDI agents*, in *Australasian conference on Computer science*. 2002, Australian Computer Society, Inc: Melbourne, Australia.
24. Wooldridge, M.J. and N.R. Jennings, *SOFTWARE ENGINEERING WITH AGENTS: Pitfalls and Pratfalls*. IEEE Internet Computing, 1999. **3**(3): p. 20-27.